# Building better software

*David Black*

Project management techniques are a disaster for software development. The alternative is “wartime software”.

“Peacetime” = predictability
“Wartime” = speed

Eg military bridge is built in 1/100th time of a civilian bridge - with higher load but “good enough” safety.

There aren’t enough people or resources – and there’s way too little time.

Velocity becomes the organizing principle.

It’s a given that your service needs to work. Without that, you lose. Once you have it you win on launching customer-value features, faster.

New rules-

**1/ Don’t break what customers already have**

No feature that may be released in the future can possibly be more important than the features that customers are using today.

**2/ Live incremental testing (rollout) & instant fallback**

You can only learn from customers and to do that you need to ship.

Integration done often, no long-term divergent branches. Your code is always “shippable”.

No “feature freeze” or “release candidate” - you release and test constantly & incrementally.

**3/ Optimize for speed**

Plan your efforts for this week, roll out, assess, repeat.

**4/ PRDs have limited value**

The only way to understand requirements is to build something and try it—with customers.

**5/ Documentation**

Only write if it helps the customers.

**6/ Risk management**

The biggest real risk in any organization is that you'll take way too much time and money delivering inappropriate stuff too late.

**7/ Iterative design**

Instead of designing things up front, your time is better spent throwing up something that gets customer feedback absolutely as quickly as you possibly can.

Don't you end up with a messy system? Yes!

And the proper thing to do is to then go back and fix things. Post-hoc design.

Make the code ready to change by eliminating redundancy- ideally one or a few places where you need to make changes. Instead of trying to anticipate where the change will be needed.

**8/ Fight entropy**

Clean up after you've shipped.

**9/ One place to change**

"Do you make the code more object-oriented? Do you apply some naming convention to the variable names? Do you segregate the code into nice, clean three-tier portions? None of the above." 😂

**10/ No separate QA environment**

Deploy. Get code in front of users. QA will use the same environment as the users.

Release small changes frequently. Have instant fallback if something goes wrong.

**11/ Replication instead of backup**

Disaster recovery plan: active-active switch.

“Backup is bad for the simple reason that the only way to use a backup is to recover it, which a time-consuming and error-prone process.”

**12/ Avoid central choke points**

Eg sharding - different dbs for different customers instead of one “multi-tenant” db.

Optimize for speed.

Web server+application+data on a single machine is simple.

**13/ Avoid “trends”**

Eg Pair programming, TDD, Agile, Cloud.

**14/ Find the right people**

Best WTS teams are way smaller than you can guess looking at their website. They are less specialized and compartmentalized.

Shipping fast attracts the best and drives away unproductive people.  Short feedback loop makes engineers happy.

Converting people from “traditional” approach to WTS is hard. The hardest part is un-learning years of hard-won skills.

Eg a six-week Perl+MySQL app that took nine months (!) to rewrite the “right way” - and they haven’t done yet. But everyone thought it’s a great idea.

**15/ Who’s got the higher status?**

Traditionally, people with the highest status are those most further from the customer.

Eg an architect- “don’t interrupt him; he’s thinking deep thoughts about the future of our code; we’d be lost without him; besides when you tell him about problems actual customers are having he gets cranky”.

Make sure your highest paid, highest status people align their actions and priorities to those of your business. They’d better be all over customers and not sitting in the glass tower designing “architecture”.

https://www.amazon.com/Wartime-Software-ebook/dp/B00CUGHDT8/